Python on Mac OS X

Session 613

These are confidential sessions—please refrain from filming or taking pictures.

Enterprise IT

Python on Mac OS X

Bill Bumgarner
bbum@apple.com
Bob Ippolito
bob@redivi.com

http://developer.apple.com/wwdc2005/

Introduction

- Python is a powerful high-level language
- Integrates extremely well with Objective-C
- You can use it to quickly develop, extend, and test applications on Mac OS X

What You'll Learn

- Why Python?
- Where is it?
- Python syntax for Objective-C programmers
- Cocoa applications in Python
- Mixed Python/Objective-C applications
- Plug-ins in Python

Why Python?

http://developer.apple.com/wwdc2005/

Python the Language

- Interpreted, interactive, object-oriented
- Powerful but clear syntax
- Excellent support for data structures
- Dynamically typed
- Garbage collection

Python on Mac OS X

- Unparalleled integration with Objective-C
- Already used in real shrink-wrapped applications
- Cross-platform (when you want it to be…)

Less Code

- Shorter development cycle
- Fewer bugs
- Better tests
- More features

Where Is It?

http://developer.apple.com/wwdc2005/

Python with Mac OS X

- Mac OS X 10.4
  - Python 2.3.5 (framework build)
  - wxPython
  - Tkinter
  - CoreGraphics (Apple-only)
  - Missing bsddb, readline

Current Python

- Python 2.4.1
  - Installer at python.org (framework build)
  - Tkinter
  - Working bsddb, readline
  - No CoreGraphics (not open source)
  - Use to create Mac OS X 10.3 compatible apps
- Installed to:
  - /Library/Frameworks/Python.framework
  - /usr/local/bin (symlinks)

Third Party Python

- pythonmac.org/packages
  - Easy to install binary packages
- Compile from source
  - Sudo python setup.py install
- Compile, package, install (requires py2app)
  - bdist_mpkg

Python Data Structures

http://developer.apple.com/wwdc2005/

list

- Mutable
- Not hashable

```
# Python
aList = ["a", "b", "c"]
aList.append("d")

// Objective-C
NSMutableArray *anArray =
    [NSMutableArray arrayWithObjects: @"a", @"b", @"c", nil];
[anArray addObject:@"d"];
```

tuple

- Immutable
- Hashable (if its contents are)

```
# Python
aTuple = ("a", "b", "c")

// Objective-C
NSArray *anArray =
    [NSArray arrayWithObjects: @"a", @"b", @"c", nil];
```

dict

- Mutable
- Not hashable
- Does not copy keys (must be hashable)

```
# Python
aDict = {
  "a": "b",
  "c": "d",
}


// Objective-C
NSMutableDictionary *aDict = [NSMutableDictionary
    dictionaryWithObjectsAndKeys: @"b", @"a", @"c", @"d", nil];
```

unicode

- Immutable
- Stores characters

```
# Python
aString = u"my string"
aCopyright = u"\xA9"
aCopyright = u"\u00A9"
aCopyright = u"\N{COPYRIGHT SIGN}"

// Objective-C
NSString *aString = @"my string";
NSString *aCopyright = [NSString stringWithUTF8String:"\xC2\xE9"];
```

str

- Immutable
- Stores bytes
  - But acts like characters (ugh)

```
# Python
aString = "what encoding is this?"
someData = "\xFE\xED\xFA\xCE"

// Objective-C
NSString *aString =
    [NSString stringWithCString:"what encoding is this?"];
NSData *someData =
    [NSData dataWithBytes:"\xFE\xED\xFA\xCE" length:4];
```

int, float

- Immutable
- Python int is a C long
- Python float is a C double

```
# Python
anInt = 100
aFloat = 3.14159

// Objective-C
long aLong = 100;
double aFloat = 3.14159;
NSNumber *anIntObject = [NSNumber numberWithLong:100];
NSNubmer *aFloatObject = [NSNumber numberWithDouble:3.14159];
```

bool

- Immutable (two singletons)
- Subclass of int
  - But it looks pretty

```
# Python
aBool = True
theOtherBool = False
```

long

- Immutable
- Has no C equivalent
- “Infinite” extent

```
# Python
aLong = 340282366920938463463374607431768211456L
```

None

- Immutable (singleton)
- Sometimes like +[NSNull null]
- Other times like nil or NULL
  - But will not receive messages

```
# Python
anObject = None
aList.append(anObject)
# WILL RAISE EXCEPTION!
anObject.doSomething()

// Objective-C
[anArray appendObject:[NSNull null]];
id anObject = nil;
[anObject doSomething];
```

Python Syntax

http://developer.apple.com/wwdc2005/

Indentation

- No more braces or semicolons
- Colons and indentation denotes blocks

```
# Python
if a == b:
    print "Python!"


// Objective-C
if ([a isEqualTo:b]) {
    NSLog(@"Objective-C");
}
```

Variables

- No type declaration
- Everything is an object

```
# Python
anInt = 1
aDict = {}


// Objective-C
int anInt = 1;
NSMutableDictionary *aDict = [NSMutableDictionary dictionary];
```

Functions

- First class objects
- Positional and/or keyword arguments

```
# Python
def printThis(aString):
    print aString


// Objective-C
void printThis(NSString *aString)
{
    NSLog(aString);
}
```

Classes

- Implementation is interface
- Methods are instance variables
- Explicit “self” in methods

```
# Python
class MyClass(MySuper):
    def __init__(self, anAttribute):
      self.anAttribute = anAttribute

    def myMethod(self, anArgument):
       pass

myObject = MyClass("attribute")
print myObject.anAttribute
```

Iteration

- for syntax only for iterators

```
# Python
for i in range(10):
  print i
for item in myList:
   print item

// Objective-C
int i;
for (i=0; i<10; i++) { ... }
NSEnumerator *anEnumerator = ...;
id item;
while ((item = anEnumerator nextObject])) {
    ...;
}
```

Item Getting

- Same syntax for list and dict

```
# Python
firstElement = aList[0]
lastElement = aList[-1]
lastElement = aList[len(aList) - 1]

keyValue = aDict["key"]


// Objective-C
id firstElement = [anArray objectAtIndex:0];
id lastElement = [anArray lastObject];
lastElement = [anArray objectAtIndex:[anArray count] - 1];

id keyValue = [aDict objectForKey:@"key"];
```

Slicing

- Can take a start, stop, and/or step

```
# Python
notTheFirst = aList[1:]
firstTwoOrLess = aList[:2]
listBackwards = aList[::-1]
listCopy = aList[:]

// Objective-C
NSArray arrayCopy = [anArray
  Imagine a lot of code here
  with NSMakeRange(...) and possibly some for() loops.
];
```

Modules (”import this”)

- Namespaces for “stuff”
- One module = One file
- Circular imports are bad

```
# Explicit namespace
import anotherModule
anotherModule.anotherFunction()

# Pick and choose
from anotherModule import anotherFunction
anotherFunction()

# Most C-like
from anotherModule import *
anotherFunction()
```

PyObjC

http://developer.apple.com/wwdc2005/

Bridge crossing

- Use Objective-C from Python
- Use Python from Objective-C
- No glue required

Selectors

- De-interleave arguments from selector
- Replace colons with underscores

```
// Objective-C
[obj setValue:@"value" forKey:@"key"];

# Python
obj.setValue_forKey_("value", "key")
```

Subclassing

- Creates a new class in both runtimes
- Same rules as Objective-C

```
# Python
class VisibleEverywhere(NSObject):
    pass

// Objective-C
Class VisibleEverywhere = NSClassFromString(@"VisibleEverywhere");
```

Conveniences

Objective-C	Python
`NSString, char*`	`unicode, str`
`NSData`	`buffer`
`NSArray`	`list, tuple`
`NSDictionary`	`dict`
`NSNumber (BOOL, int, …)`	`int, long, float, …`
`NSDecimalNumber`	`<custom number-like wrapper>`
`FSRef`	`Carbon.File.FSRef`
…and more!	

Wrapped Frameworks

- Apple frameworks wrapped out of the box
  - AppKit
  - Foundation
  - CoreData
  - (and more!)
- NSBundle API works too
- Or use objc.loadBundle()

Xcode Templates

- Lots of magic
- Development and Deployment are targets
- Uses py2app to build
  - Takes care of all Python dependencies
  - Automatically finds dependent frameworks
  - Ensures the Mach-O headers have bundle-relative install names

Demo

CoreData

http://developer.apple.com/wwdc2005/

PyObjC Mixed Applications

http://developer.apple.com/wwdc2005/

Demo

ReSTedit

http://developer.apple.com/wwdc2005/

PyObjC Plugins

http://developer.apple.com/wwdc2005/

Demo

PyInterpreter IBPalette

http://developer.apple.com/wwdc2005/

Embedding Python

http://developer.apple.com/wwdc2005/

Demo

CodeImporter

http://developer.apple.com/wwdc2005/

For More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Q&A

Bill Bumgarner
bbum@apple.com
Bob Ippolito
bob@redivi.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference